basename()

Vulnerable to path spoofing

Sean Barnum, Cigital, Inc. [vita[1]]

Copyright © 2005 Cigital, Inc.

2005-10-03

##### "Original Cigital Coding Rule in XML"

Mime-type: text/xml, ######: 7861 bytes

Identification Difficulty

Scan

Rule Accuracy

False Positives

Priority

Medium

Attack Categories

- Path spoofing or confusion problem

Vulnerability Categories

- Indeterminate File/Path
- TOCTOU - Time of Check, Time of Use

Software Context

Filename Management File Path Management

Description

Note: dirname, basename functions should be analyzed together The basename() function returns the last component from the pathname pointed to by path, deleting any trailing "/" characters. If path consists entirely of "/" characters, a pointer to the string "/" is returned. If path is a NULL pointer or the empty string, a pointer to the string "." is returned. basename() is vulnerable to path spoofing

[1] daisy:35 (Barnum, Sean)

Application Programming Interfaces

Function Name	Comments
basename	
dirname	

Method of Attack

The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows an attacker to change the state of the targeted resource and yield unexpected and undesired results. The dirname() call is a use-category call, which when preceded by a check-category call can be indicative of a TOCTOU vulnerability. A TOCTOU attack in regards to basename() can occur when

1. A check for the existence of a file or other reference to a file/directory name, to be parsed by basename, occurs
2. The basename call is executed to return the pathname of the parent directory of the directory/filename

Between 1 and 2, an attacker could, for example, link the target file (the file to be parsed) to a different known file. The subsequent parse of the target file would result in, at a minimum, potentially erroneous program function.

Solutions

Applicability	Description	Efficacy
Generally applies to any basename() call.	Translate basename() into function(s) using file descriptors, if possible.	Effective.
Generally applicable.	The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but it does help to limit the false sense of security given by the check.	Does not resolve the underlying vulnerability but limits the false sense of security given by the check.
Generally applicable.	Limit the interleaving of operations on files from multiple processes.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
Generally applicable.	Limit the spread of time (cycles)	Does not eliminate the

	between the check and use of a resource.	underlying vulnerability but can help make it more difficult to exploit.
Generally applicable.	Attempt the basename and then perform an application-specific sanity check after the basename call.	Checking the status after the operation does not change the fact that the operation may have been exploited but it does allow halting of the application in an error state to help limit further damage.

Signature Details

char *basename(char *path);

Examples of Incorrect Code

- Example 1

```
int main()
{
  struct stat stats;
  char *path="/etc/passwd";
  char *bname;

  stat(path, &stats);
  bname=basename(path);
  return 0;
}
```

Examples of Corrected Code

- Example 1

```
int main()
{
  char *path="/etc/passwd";
  char *bname;

  bname=basename(path);

  /* This does not eliminate the vulnerability but does remove
     the false sense of security brought by the check call */

  /* An application-specific sanity check needs to occur
     at this point to verify correct behavior occurred*/

  return 0;
}
```

Source References

- Viega, John & McGraw, Gary. Building Secure Software: How to Avoid Security Problems the Right Way. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, ch 9.
- Howard, Michael & LeBlanc, David C. Writing Secure Code, 2nd ed. Redmond, WA: Microsoft Press, 2002, ISBN: 0735617228.

- Using the Strsafe.h Functions - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/resources/strings/usingstrsafefunctions.asp[2]

Recommended Resources

Resource	Link
UNIX man page for basename()	http://www.freebsd.org/cgi/man.cgi?query=basename&sektion=3[3]

Discriminant Set

Operating Systems

- UNIX Windows

Languages

- C
- C++

Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005. Cigital-authored documents are sponsored by the U.S. Department of Defense under Contract FA8721-05-C-0003. Cigital retains copyrights in all material produced under this contract. The U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce these documents, or allow others to do so, for U.S. Government purposes only pursuant to the copyright license under the contract clause at 252.227-7013.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Cigital, including information about “Fair Use,” contact Cigital at copyright@cigital.com[1].

####

###	########
Copyright Holder	Cigital, Inc.

####

###	########

[2] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/resources/strings/usingstrsafefunctions

[3] http://www.freebsd.org/cgi/man.cgi?query=basename&sektion=3

[1] mailto:copyright@cigital.com

Attack Categories	Path Spoofing or Confusion
Operating System	UNIX Windows
Software Context	File Management File Path Management
Vulnerability Categories	Indeterminate File/Path Time-of-Check/Time-of-Use